A Polynomial-Time Parsing Algorithm for TT-MCTAG

Laura Kallmeyer
Collaborative Research Center 441
Universität Tübingen
Tübingen, Germany
lk@sfs.uni-tuebingen.de

Giorgio Satta
Department of Information Engineering
University of Padua
Padova, Italy
satta@dei.unipd.it

Abstract

This paper investigates the class of Tree-Tuple MCTAG with Shared Nodes, TT-MCTAG for short, an extension of Tree Adjoining Grammars that has been proposed for natural language processing, in particular for dealing with discontinuities and word order variation in languages such as German. It has been shown that the universal recognition problem for this formalism is NP-hard, but so far it was not known whether the class of languages generated by TT-MCTAG is included in PTIME. We provide a positive answer to this question, using a new characterization of TT-MCTAG.

1 Introduction

For a large range of linguistic phenomena, extensions of Tree Adjoining Grammars (Joshi et al., 1975), or TAG for short, have been proposed based on the idea of separating the contribution of a lexical item into several components. Instead of single trees, these grammars contain (multi-)sets of trees. Examples are tree-local and set-local multicomponent TAG (Joshi, 1985; Weir, 1988), MCTAG for short, non-local MCTAG with dominance links (Becker et al., 1991), Vector-TAG with dominance links (Rambow, 1994) and, more recently, Tree-Tuple MCTAG with Shared Nodes (Lichte, 2007)), or TT-MCTAG for short.

For some of the above formalisms the word recognition problem is NP-hard. This has been shown for non-local MCTAG (Rambow and Satta, 1992), even in the lexicalized case (Champollion, 2007). Some others generate only polynomial languages but their generative capacity is too limited to deal with all natural language phenomena. This has been argued for tree-local and even set-local MCTAG on the basis of scrambling data from languages such as German (Becker et al., 1992; Rambow, 1994).

In this paper, we focus on TT-MCTAG (Lichte, 2007). So far, it has been shown that the universal recognition problem for TT-MCTAG is NP-hard (Søgaard et al., 2007). A restriction on TT-MCTAG has been proposed in (Kallmeyer and Parmentier, 2008): with such a restriction, the universal recognition problem is still NP-hard, but the class of generated languages is included in PTIME, i.e., all these languages can be recognized in deterministic polynomial time. In this paper, we address the question of whether for general TT-MCTAG, i.e., TT-MCTAG without the constraint from (Kallmeyer and Parmentier, 2008), the class of generated languages is included in PTIME. We provide a positive answer to this question.

The TT-MCTAG definition from (Lichte, 2007; Kallmeyer and Parmentier, 2008) imposes a condition on the way different tree components from a tree tuple in the grammar combine with each other. This condition is formulated in terms of mapping between argument and head trees, i.e., in order to test such a condition one has to guess some grouping of the tree components used in a derivation into instances of tree tuples from the grammar. This results in a combinatorial explosion of parsing analyses. In order to obtain a polynomial parsing algorithm, we need to avoid this effect.

On this line, we propose an alternative characterization of TT-MCTAG that only requires (i) a counting of tree components and (ii) the check of some local conditions on these counts. This allows for parsing in polynomial deterministic time.

TT-MCTAG uses so-called ‘parallel unordered’ rewriting. The first polynomial time parsing results on this class were presented in (Rambow and Satta, 1994; Satta, 1995) for some string-based systems, exploiting counting techniques closely related to those we use in this paper. In contrast to string-based rewriting, the tree rewriting formalisms we consider here are structurally more complex and require specializations of the above techniques. Polynomial parsing results for tree rewriting systems based on parallel unordered rewriting have also been reported in (Rambow, 1994; Rambow et al., 1995). However, in the approach proposed by these authors, tree-based grammars are first translated into equivalent string-based systems, and the result is again provided on the string domain.

2 Tree Adjoining Grammars

Tree Adjoining Grammars (Joshi et al., 1975) are a formalism based on tree rewriting. We briefly summarize here the relevant definitions and refer the reader to (Joshi and Schabes, 1997) for a more complete introduction.

Definition 1 A **Tree Adjoining Grammar** (TAG) is a tuple $G = (V_N, V_T, S, I, A)$ where V_N and V_T are disjoint alphabets of non-terminal and terminal symbols, respectively, $S \in V_N$ is the start symbol, and I and A are finite sets of **initial** and **auxiliary** trees, respectively. □

Trees in $I \cup A$ are called **elementary** trees. The internal nodes in the elementary trees are labeled with non-terminal symbols, the leaves with non-terminal or terminal symbols. As a special property, each auxiliary tree β has exactly one of its leaf nodes marked as the **foot** node, having the same label as the root. Such a node is denoted by $Ft(\beta)$. Leaves with non-terminal labels that are not foot nodes are called **substitution** nodes.

In a TAG, larger trees can be derived from the elementary trees by subsequent applications of the operations substitution and adjunction. The **substitution** operation replaces a substitution node η with an initial tree having root node with the same label as η. The **adjunction** operation replaces an internal node η in a previously derived tree γ with an auxiliary tree β having root node with the same label as η. The subtree of γ rooted at η is then placed below the foot node of β. Only internal nodes can allow for adjunction, adjunction at leaves is not possible. See figure 1 for an example of a tree derivation.

Usually, a TAG comes with restrictions on the two operations, specified at each node η by sets $Sbst(\eta)$ and $Adj(\eta)$ listing all elementary trees that can be substituted or adjoined, respectively. Furthermore, adjunction at η might be obligatory.

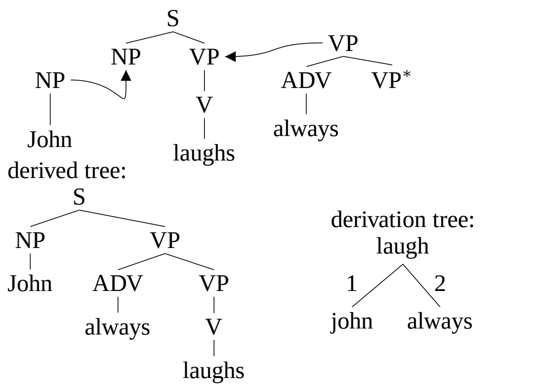

Figure 1: TAG derivation for *John always laughs*

TAG derivations are represented by derivation trees that record the history of how the elementary trees are put together. A **derivation tree** is an unordered tree whose nodes are labeled with elements in $I \cup A$ and whose edges are labeled with Gorn addresses of elementary trees.[1] Each edge in a derivation tree stands for an adjunction or a substitution. E.g., the derivation tree in figure 1 indicates that the elementary tree for *John* is substituted for the node at address 1 and *always* is adjoined at node address 2.

In the following, we write a derivation tree D as a directed graph $\langle V, E, r \rangle$ where V is the set of nodes, $E \subset V \times V$ is the set of arcs and $r \in V$ is the root. For every $v \in V$, $Lab(v)$ gives the node label and for every $\langle v_1, v_2 \rangle \in E$, $Lab(\langle v_1, v_2 \rangle)$ gives the edge label.

A **derived tree** is the result of carrying out the substitutions and the adjunctions in a derivation tree, i.e., the derivation tree describes uniquely the derived tree; see again figure 1.

3 TT-MCTAG

3.1 Introduction to TT-MCTAG

For a range of linguistic phenomena, multicomponent TAG (Weir, 1988) have been proposed, also called MCTAG for short. The underlying motivation is the desire to split the contribution of a single lexical item (e.g., a verb and its arguments) into several elementary trees. An MCTAG consists of (multi-)sets of elementary trees, called **tree sets**. If an elementary tree from some set is used in a derivation, then all of the remaining trees in the set must be used as well. Several variants of MCTAGs can be found the literature, differing on the

[1]In this convention, the root address is ε and the jth child of a node with address p has address $p \cdot j$.

specific definition of the derivation process.

The particular MCTAG variant we are concerned with is Tree-Tuple MCTAG with Shared Nodes, TT-MCTAG (Lichte, 2007). TT-MCTAG were introduced to deal with free word order phenomena in languages such as German. An example is (1) where the argument *es* of *reparieren* precedes the argument *der Mann* of *versucht* and is not adjacent to the predicate it depends on.

(1)	... dass	es	der Mann	zu reparieren	versucht
	... that	it	the man	to repair	tries
	'... that the man tries to repair it'				

A TT-MCTAG is slightly different from standard MCTAGs since each elementary tree set contains one specially marked lexicalized tree called the head, and all of the remaining trees in the set function as arguments of the head. Furthermore, in a TT-MCTAG derivation the argument trees must either adjoin directly to their head tree, or they must be linked in the derivation tree to an elementary tree that attaches to the head tree, by means of a chain of adjunctions at root nodes. In other words, in the corresponding TAG derivation tree, the head tree must dominate the argument trees in such a way that all positions on the path between them, except the first one, must be labeled by ε. This captures the notion of adjunction under node sharing from (Kallmeyer, 2005).[2]

Definition 2 A **TT-MCTAG** is a tuple $G = (V_N, V_T, S, I, A, \mathcal{T})$ where $G_T = (V_N, V_T, S, I, A)$ is an underlying TAG and $\mathcal{T}$ is a finite set of **tree tuples** of the form $\Gamma = \langle\gamma, \{\beta_1, \dots, \beta_r\}\rangle$ where $\gamma \in (I \cup A)$ has at least one node with a terminal label, and $\beta_1, \dots, \beta_n \in A$. □

For each $\Gamma = \langle\gamma, \{\beta_1, \dots, \beta_r\}\rangle \in \mathcal{T}$, we call γ the **head tree** and the β_j's the **argument trees**. We informally say that γ and the β_j's belong to Γ, and write $|\Gamma| = r + 1$.

As a remark, an elementary tree γ from the underlying TAG G_T can be found in different tree tuples in G, or there could even be multiple instances of such a tree within the same tree tuple Γ. In these cases, we just treat these tree instances as distinct trees that are isomorphic and have identical labels.

[2]The intuition is that, if a tree γ' adjoins to some γ, its root in the resulting derived tree somehow belongs both to γ and γ' or, in other words, is shared by them. A further tree β adjoining to this node can then be considered as adjoining to γ, not only to γ' as in standard TAG. Note that we assume that foot nodes do not allow adjunctions, otherwise node sharing would also apply to them.

For a given argument tree β in Γ, $h(\beta)$ denotes the head of β in Γ. For a given $\gamma \in I \cup A$, $a(\gamma)$ denotes the set of argument trees of γ, if there are any, or the empty set otherwise. Furthermore, for a given TT-MCTAG G, $H(G)$ is the set of head trees and $A(G)$ is the set of argument trees. Finally, a node v in a derivation tree for G with $Lab(v) = \gamma$ is called a γ-node.

Definition 3 Let $G = (V_N, V_T, S, I, A, \mathcal{T})$ be some TT-MCTAG. A derivation tree $D = \langle V, E, r\rangle$ in the underlying TAG G_T is licensed in G if and only if the following conditions (MC) and (SN-TTL) are both satisfied.

- **(MC)**: For all Γ from G and for all γ_1, γ_2 in Γ, we have $|\{v \mid v \in V,\ Lab(v) = \gamma_1\}| = |\{v \mid v \in V,\ Lab(v) = \gamma_2\}|$.
- **(SN-TTL)**: For all $\beta \in A(G)$ and $n \geq 1$, let $v_1, \dots, v_n \in V$ be pairwise different $h(\beta)$-nodes, $1 \leq i \leq n$. Then there are pairwise different β-nodes $u_1, \dots, u_n \in V$, $1 \leq i \leq n$. Furthermore, for $1 \leq i \leq n$, either $\langle v_i, u_i\rangle \in E$, or else there are $u_{i,1}, \dots, u_{i,k}$, $k \geq 2$, with auxiliary tree labels, such that $u_i = u_{i,k}$, $\langle v_i, u_{i,1}\rangle \in E$ and, for $1 \leq j \leq k-1$, $\langle u_{i,j}, u_{i,j+1}\rangle \in E$ with $Lab(\langle u_{i,j}, u_{i,j+1}\rangle) = \varepsilon$. □

The separation between (MC) and (SN-TTL) in definition 3 is motivated by the desire to separate the multicomponent property that TT-MCTAG shares with a range of related formalisms (e.g., tree-local and set-local MCTAG, Vector-TAG, etc.) from the notion of tree-locality with shared nodes that is peculiar to TT-MCTAG.

Figure 2 shows a TT-MCTAG derivation for (1). Here, the NP$_{nom}$ auxiliary tree adjoins directly to *versucht* (its head) while the NP$_{acc}$ tree adjoins to the root of a tree that adjoins to the root of a tree that adjoins to *reparieren*.

TT-MCTAG can generate languages that, in a strong sense, cannot be generated by Linear Context-Free Rewriting Systems (Vijay-Shanker et al., 1987; Weir, 1988), or LCFRS for short. An example is the language of all strings $\pi(n^{[1]} \dots n^{[m]})v^{[1]} \dots v^{[m]}$ with $m \geq 1$, π a permutation, and $n^{[i]} = n$ is a nominal argument of $v^{[i]} = v$ for $1 \leq i \leq m$, i.e., these occurrences come from the same tree set in the grammar. Such a language has been proposed as an abstract description of the scrambling phenomenon as found in German and other free word order languages,

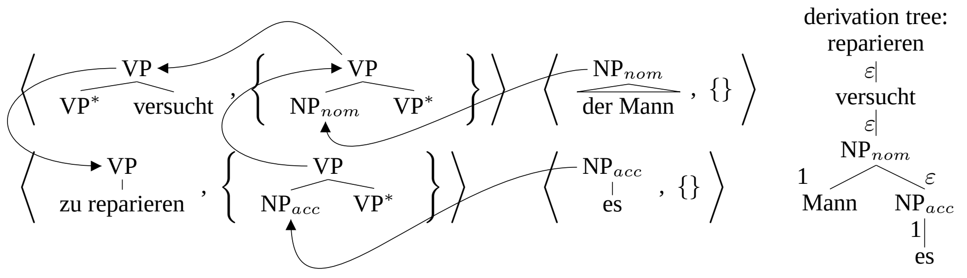

Figure 2: TT-MCTAG derivation of (1)

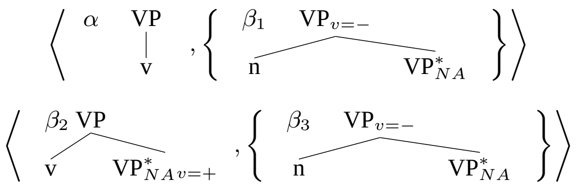

Figure 3: TT-MCTAG

and cannot be generated by a LCFRS (Becker et al., 1992; Rambow, 1994). Figure 3 reports a TT-MCTAG for this language.

Concerning the other direction, at the time of writing it is not known whether there are languages generated by LCFRS but not by TT-MCTAG. It is well known that LCFRS is closed under the finite-copy operator. This means that, for any fixed $k > 1$, if L is generated by a LCFRS then the language $\{w \,|\, w = u^k,\ u \in L\}$ can also be generated by a LCFRS. We conjecture that TT-MCTAG does not have such a closure property. However, from a first inspection of the MC-TAG analyses proposed for natural languages (see Chen-Main and Joshi (2007) for an overview), it seems that there are no important natural language phenomena that can be described by LCFRS and not by TT-MCTAG. Any construction involving some kind of component stacking along the VP projection such as subject-auxiliary inversion can be modelled with TT-MCTAG. Unbounded extraposition phenomena cannot be described with TT-MCTAG but they constitute a problem for any local formalism and so far the nature of these phenomena is not sufficiently well-understood.

Note that, in contrast to non-local MCTAG, in TT-MCTAG the trees coming from the same instance of a tuple in the grammar are not required to be added at the same time. TT-MCTAGs share this property of 'non-simultaneity' with other vector grammars such as Unordered Vector Grammars (Cremers and Mayer, 1973) and Vector-TAG (Rambow, 1994), V-TAG for short, and it is crucial for the polynomial parsing algorithm. The non-simultaneity seems to be an advantage when using synchronous grammars to model the syntax-semantics interface (Nesson and Shieber, 2008). The closest formalism to TT-MCTAG is V-TAG. However, there are fundamental differences between the two. Firstly, they make a different use of dominance links: In V-TAG dominance links relate different nodes in the trees of a tree set from the grammar. They present dominance requirements that constrain the derived tree. In TT-MCTAG, there are no dominance links between nodes in elementary trees. Instead, the node of a head tree in the derivation tree must dominate all its arguments. Furthermore, even though TT-MCTAG arguments can adjoin with a delay to their head, their possible adjunction site is restricted with respect to their head. As a result, one obtains a slight degree of locality that can be exploited for natural language phenomena that are unbounded only in a limited domain. This is proposed in (Lichte and Kallmeyer, 2008) where the fact that substitution nodes block argument adjunction to higher heads is used to model the limited domain of scrambling in German. V-TAG does not have any such notion of locality. Instead, it uses explicit constraints, so-called integrity constraints, to establish islands.

3.2 An alternative characterization of TT-MCTAG

The definition of TT-MCTAG in subsection 3.1 is taken from (Lichte, 2007; Kallmeyer and Parmentier, 2008). The condition (SN-TTL) on the TAG derivation tree is formulated in terms of heads and arguments belonging together, i.e., coming from the same tuple instance. For our parsing algorithm, we want to avoid grouping the instances of elementary trees in a derivation tree into tuple instances. In other words, we want to check whether a TAG derivation tree is a valid TT-

MCTAG derivation tree without deciding, for every occurrence of some argument β, which of the $h(\beta)$-nodes represents its head. Therefore we propose to reformulate (SN-TTL).

For a node v in a derivation tree D, we write D_v to represent the subtree of D rooted at v. For $\gamma \in (I \cup A)$, we define $Dom(v, \gamma)$ as the set of nodes of D_v that are labeled by γ. Furthermore, for an argument tree $\beta \in A(G)$, we let $\pi(v, \beta) = |Dom(v, \beta)| - |Dom(v, h(\beta))|$.

Lemma 1 *Let G be a TT-MCTAG with underlying TAG G_T, and let $D = \langle V, E, r\rangle$ be a derivation tree in G_T that satisfies (MC). D satisfies (SN-TTL) if and only if, for every $v \in V$ and every $\beta \in A(G)$, the following conditions both hold.*

(i) $\pi(v, \beta) \geq 0$.

(ii) *If* $\pi(v, \beta) > 0$, *then one of the following conditions must be satisfied:*

   (a) $Lab(v) = \beta$ *and* $\pi(v, \beta) = 1$*;*

   (b) $Lab(v) = \beta$ *and* $\pi(v, \beta) > 1$*, and there is some* $\langle v, v_\varepsilon\rangle \in E$ *with* $Lab(\langle v, v_\varepsilon\rangle) = \varepsilon$ *and* $\pi(v_\varepsilon, \beta) + 1 = \pi(v, \beta)$*;*

   (c) $Lab(v) \notin \{\beta, h(\beta)\}$ *and there is some* $\langle v, v_\varepsilon\rangle \in E$ *with* $Lab(\langle v, v_\varepsilon\rangle) = \varepsilon$ *and* $\pi(v_\varepsilon, \beta) = \pi(v, \beta)$*;*

   (d) $Lab(v) = h(\beta)$ *and there is some* $\langle v, v_\varepsilon\rangle \in E$ *with* $Lab(\langle v, v_\varepsilon\rangle) = \varepsilon$ *and* $\pi(v, \beta) \leq \pi(v_\varepsilon, \beta) \leq \pi(v, \beta) + 1$.

Intuitively, condition (i) in lemma 1 captures the fact that heads always dominate their arguments in the derivation tree. Condition (ii)b states that, if v is a β-node and if v is not the only 'pending' β-node in D_v, then all pending β-nodes in D_v, except v itself, must be below the root adjoining node. Here pending means that the node is not matched to a head-node within D_v. Condition (ii)c treats the case in which there are pending β-nodes in D_v for some node v whose label is neither β nor $h(\beta)$. Then the pending nodes must all be below the root adjoining node. Finally, condition (ii)d deals with the case of a $h(\beta)$-node v where, besides the β-node that serves as an argument of v, there are other pending β-nodes in D_v. These other pending β-nodes must all be in D_{v_ε}, where v_ε is the (unique) root adjoining node, if it exists. The argument of v might as well be below v_ε, and then the number of pending β-nodes in D_{v_ε} is the number of pending nodes in D_v, incremented by 1, since the argument of v is not pending in D_v but it is pending in D_{v_ε}. Otherwise, the argument of v is a pending β-node below some other daughter of v. Then the number of pending β-nodes in D_{v_ε} is the same as in D_v.

PROOF We first show that (SN-TTL) implies both (i) and (ii).

Condition (i): Assume that there is a $v \in V$ and a $\beta \in A(G)$ with $\pi(v, \beta) < 0$. Then for some n and for pairwise different $v_1, \ldots, v_n$ with $\langle v, v_i\rangle \in E^*$, $Lab(v_i) = h(\beta)$ $(1 \leq i \leq n)$, we cannot find pairwise different $u_1, \ldots, u_n$ with $\langle v_i, u_i\rangle \in E^*$, $Lab(u_i) = \beta$. This is in contradiction with (SN-TTL). Consequently, condition (i) must be satisfied.

Condition (ii): Assume β and v as in the statement of the lemma, with $\pi(v, \beta) > 0$. Let $v_1, \ldots, v_n$ be all the $h(\beta)$-nodes in D. There is a bijection f_β from these nodes to n pairwise distinct β-nodes in D, such that every pair v_i, $f_\beta(v_i) = u_i$ satisfies the conditions in (SN-TTL). Because of (MC), the nodes $u_1, \ldots, u_n$ must be all the β-nodes in D. There must be at least one v_i $(1 \leq i \leq n)$ with $\langle v_i, v\rangle \in E^+$, $\langle v, f_\beta(v_i)\rangle \in E^*$. Then we have one of the following cases.

(a) $u_i = v$ and v_i is the only $h(\beta)$-node dominating v with a corresponding β-node dominated by v. In this case (ii)a holds.

(b) $Lab(v) = \beta$, i.e., $\langle f_\beta^{-1}(v), v\rangle \in E^+$ and there are other nodes $u \in Dom(v, \beta)$, $u \neq v$ with $\langle f_\beta^{-1}(u), v\rangle \in E^+$. Then, with (SN-TTL), there must be a v_ε with $\langle v, v_\varepsilon\rangle \in E$, $Lab(\langle v, v_\varepsilon\rangle) = \varepsilon$ and for all such nodes u, $\langle v_\varepsilon, u\rangle \in E^*$. Consequently, (ii)b holds.

(c) $Lab(v) \notin \{\beta, h(\beta)\}$. Then, as in (b), there must be a v_ε with $\langle v, v_\varepsilon\rangle \in E$, $Lab(\langle v, v_\varepsilon\rangle) = \varepsilon$ and for all $u \in Dom(v, \beta)$ with $\langle f_\beta^{-1}(u), v\rangle \in E^+$, $\langle v_\varepsilon, u\rangle \in E^*$. Consequently, (ii)c holds.

(d) $Lab(v) = h(\beta)$. If $f_\beta(v)$ is dominated by a v_ε that is a daughter of v with $Lab(\langle v, v_\varepsilon\rangle) = \varepsilon$, then for all $u \in Dom(v, \beta)$ with $\langle f_\beta^{-1}(u), v\rangle \in E^+$ we have $\langle v_\varepsilon, u\rangle \in E^*$. Consequently, $\pi(v_\varepsilon, \beta) = \pi(v, \beta) + 1$. Alternatively, $f_\beta(v)$ is dominated by some other daughter v' of v with $Lab(\langle v, v'\rangle) \neq \varepsilon$. In this case v_ε must still exist and, for all $u \in Dom(v, \beta)$ with $u \neq f_\beta(v)$ and with $\langle f_\beta^{-1}(u), v\rangle \in E^+$, we have $\langle v_\varepsilon, u\rangle \in E^*$. Consequently, $\pi(v_\varepsilon, \beta) = \pi(v, \beta)$.

Now we show that (i) and (ii) imply (SN-TTL). With (MC), the number of β-nodes and $h(\beta)$-nodes in V are the same, for every $\beta \in A(G)$. For every $\beta \in A(G)$, we construct a bijection f_β of the

same type as in the first part of the proof, and show that (SN-TTL) is satisfied. To construct f_β, for every $v \in V$ we define sets $V_{\beta,v} \subseteq Dom(v,\beta)$ of β-nodes v_β that have a matching head $f_\beta(v_\beta)$ dominating v. The definition satisfies $|V_{\beta,v}| = \pi(v,\beta)$. For every v with $v_1,\ldots,v_n$ being all its daughters:
a) If $Lab(v) = \beta$, then (by (ii)) for every $1 \le j \le n$ with $Lab(\langle v, v_j\rangle) \ne \varepsilon$, $V_{\beta,v_j} = \emptyset$. If there is a v_i with $Lab(\langle v,v_i\rangle) = \varepsilon$, then $V_{\beta,v} = V_{\beta,v_i} \cup \{v\}$, else $V_{\beta,v} = \{v\}$.
b) If $Lab(v) \notin \{\beta, h(\beta)\}$, then (by (ii)) $V_{\beta,v_j} = \emptyset$ for every $1 \le j \le n$ with $Lab(\langle v,v_j\rangle) \ne \varepsilon$. If there is a v_i with $Lab(\langle v,v_i\rangle) = \varepsilon$, then $V_{\beta,v} = V_{\beta,v_i}$, else $V_{\beta,v} = \emptyset$.
c) If $Lab(v) = h(\beta)$, then there must be some i, $1 \le i \le n$, such that $V_{\beta,v_i} \ne \emptyset$. We need to distinguish two cases. In the first case we have $Lab(\langle v,v_i\rangle) \ne \varepsilon$, $|V_{\beta,v_i}| = 1$ and, for every $1 \le j \le n$ with $j \ne i$, either $V_{\beta,v_j} = \emptyset$ or $Lab(\langle v,v_j\rangle) = \varepsilon$. In this case we define $f_\beta(v) = v'$ for $\{v'\} = V_{\beta,v_i}$. In the second case we have $Lab(\langle v,v_i\rangle) = \varepsilon$ and, for every $1 \le j \le n$ with $j \ne i$, $V_{\beta,v_j} = \emptyset$. In this case we pick an arbitrary $v' \in V_{\beta,v_i}$ and let $f_\beta(v) = v'$. In both cases we let $V_{\beta,v} = (\bigcup_{i=1}^{n} V_{\beta,v_i}) \setminus \{f_\beta(v)\}$.

With this mapping, (SN-TTL) is satisfied when choosing for each $h(\beta)$-node v_i the β-node $u_i = f_\beta(v_i)$ as its corresponding node. ■

4 Parsing algorithm

In this section we present a recognition algorithm for TT-MCTAG working in polynomial time in the size of the input string. The algorithm can be easily converted into a parsing algorithm. The basic idea is to use a parsing algorithm for TAG, and impose on-the-fly additional restrictions on the underlying derivation trees that are being constructed, in order to fulfill the definition of valid TT-MCTAG derivation. To simplify the presentation, we assume without loss of generality that all elementary trees in our grammars are binary trees. The input string has the form $w = a_1 \cdots a_n$ with each $a_i \in V_T$ and $n \ge 0$ ($n = 0$ means $w = \varepsilon$).

4.1 TAG recognition

We start with the discussion of a baseline recognition algorithm for TAG, along the lines of (Vijay-Shanker and Joshi, 1985). The algorithm is specified by means of deduction rules, following (Shieber et al., 1995), and can be implemented using standard tabular techniques. Items have the form $[\gamma, p_t, i, f_1, f_2, j]$ where $\gamma \in I \cup A$, p is the address of a node in γ, subscript $t \in \{\top, \bot\}$ specifies whether substitution or adjunction has already taken place ($\top$) or not ($\bot$) at p, and $0 \le i \le f_1 \le f_2 \le j \le n$ are indices with i, j indicating the left and right edges of the span recognized by p and f_1, f_2 indicating the span of a gap in case a foot node is dominated by p. We write $f_1 = f_2 = -$ if no gap is involved. For combining indices, we use the operator $f' \oplus f'' = f$ where $f = f'$ if $f'' = -$, $f = f''$ if $f' = -$, and f is undefined otherwise. The deduction rules are shown in figure 4.

The algorithm walks bottom-up on the derivation tree. Rules (1) and (2) process leaf nodes in elementary trees and require precondition $Lab(\gamma,p) = w_{i+1}$ and $Lab(\gamma,p) = \varepsilon$, respectively. Rule (3) processes the foot node of auxiliary tree $\beta \in A$ by guessing the portion of w spanned by the gap. Note that we use $p_\top$ in the consequent item in order to block adjunction at foot nodes, as usually required in TAG.

We move up along nodes in an elementary tree by means of rules (4) and (5), depending on whether the current node has no sibling or has a single sibling, respectively.

Rule (6) substitutes initial tree α at p in γ, under the precondition $\alpha \in Sbst(\gamma,p)$. Similarly, rule (7) adjoins auxiliary tree β at p in γ, under the precondition $\beta \in Adj(\gamma,p)$. Both these rules use $p_\top$ in the consequent item in order to block multiple adjunction or substitution at p, as usually required in TAG. Rule (8) processes nodes at which adjunction is not obligatory.

The algorithm recognizes w if and only if some item $[\alpha, \varepsilon_\top, 0, -, -, n]$ can be inferred with $\alpha \in I$ and $Lab(\alpha,\varepsilon) = S$.

4.2 TT-MCTAG recognition

We now extend the recognition algorithm of figure 4 to TT-MCTAG. Let G be an input TT-MCTAG. We assume that the tuples in $\mathcal{T}$ are numbered from 1 to $|\mathcal{T}|$, and that the elementary trees in each Γ_i are also numbered from 1 to $|\Gamma_i|$, with the first element being the head. We then write $\gamma_{q,r}$ for the r-th elementary tree in the q-th tuple in $\mathcal{T}$.

A **t-counter** is a ragged array T of integers with primary index q ranging over $\{1,\ldots,|\mathcal{T}|\}$ and with secondary index r ranging over $\{1,\ldots,|\Gamma_i|\}$. We write $T^{(q,r)}$ to denote the t-counter with $T[q,r] = 1$ and zero everywhere else. We also use the sum and the difference of t-counters, which are

$$\frac{}{[\gamma, p_\perp, i, -, -, i+1]} \quad (1)$$

$$\frac{}{[\gamma, p_\perp, i, -, -, i]} \quad (2)$$

$$\frac{}{[\beta, Ft(\beta)_\top, i, i, j, j]} \quad (3)$$

$$\frac{[\gamma, (p \cdot 1)_\top, i, f_1, f_2, j]}{[\gamma, p_\perp, i, f_1, f_2, j]} \quad (4)$$

$$\frac{\begin{array}{c}[\gamma, (p \cdot 1)_\top, i, f_1, f_2, k]\\ [\gamma, (p \cdot 2)_\top, k, f'_1, f'_2, j]\end{array}}{[\gamma, p_\perp, i, f_1 \oplus f'_1, f_2 \oplus f'_2, j]} \quad (5)$$

$$\frac{[\alpha, \varepsilon_\top, i, -, -, j]}{[\gamma, p_\top, i, -, -, j]} \quad (6)$$

$$\frac{\begin{array}{c}[\beta, \varepsilon_\top, i, f_1, f_2, j]\\ [\gamma, p_\perp, f_1, f'_1, f'_2, f_2]\end{array}}{[\gamma, p_\top, i, f'_1, f'_2, j]} \quad (7)$$

$$\frac{[\gamma, p_\perp, i, f_1, f_2, j]}{[\gamma, p_\top, i, f_1, f_2, j]} \quad (8)$$

Figure 4: A baseline recognition algorithm for TAG. Rule preconditions and goal item are described in the text.

$$\frac{}{[\gamma_{q,r}, p_\perp, i, -, -, i+1, T^{(q,r)}]} \quad (9)$$

$$\frac{}{[\gamma_{q,r}, p_\perp, i, -, -, i, T^{(q,r)}]} \quad (10)$$

$$\frac{}{[\gamma_{q,r}, Ft(\gamma_{q,r})_\top, i, i, j, j, T^{(q,r)}]} \quad (11)$$

$$\frac{[\gamma_{q,r}, (p \cdot 1)_\top, i, f_1, f_2, j, T]}{[\gamma_{q,r}, p_\perp, i, f_1, f_2, j, T]} \quad (12)$$

$$\frac{\begin{array}{c}[\gamma_{q,r}, (p \cdot 1)_\top, i, f_1, f_2, k, T_1]\\ [\gamma_{q,r}, (p \cdot 2)_\top, k, f'_1, f'_2, j, T_2]\end{array}}{[\gamma_{q,r}, p_\perp, i, f_1 \oplus f'_1, f_2 \oplus f'_2, j, T_1 + T_2 - T^{(q,r)}]} \quad (13)$$

$$\frac{[\gamma_{q',r'}, \varepsilon_\top, i, -, -, j, T']}{[\gamma_{q,r}, p_\top, i, -, -, j, T' + T^{(q,r)}]} \quad (14)$$

$$\frac{\begin{array}{c}[\gamma_{q',r'}, \varepsilon_\top, i, f_1, f_2, j, T']\\ [\gamma_{q,r}, p_\perp, f_1, f'_1, f'_2, f_2, T]\end{array}}{[\gamma_{q,r}, p_\top, i, f'_1, f'_2, j, T + T']} \quad (15)$$

$$\frac{[\gamma, p_\perp, i, f_1, f_2, j, T]}{[\gamma, p_\top, i, f_1, f_2, j, T]} \quad (16)$$

Figure 5: A recognition algorithm for TT-MCTAG. Rule preconditions are the same as for figure 4, filtering conditions on rules are described in the text.

defined elementwise in the obvious way.

Let D be a derivation tree generated by the TAG underlying G. We associate D with the t-counter T such that $T[q, r]$ equals the count of all occurrences of elementary tree $\gamma_{q,r}$ appearing in D. Intuitively, we use t-counters to represent information about TAG derivation trees that are relevant to the licensing of such trees by the input TT-MCTAG G.

We are now ready to present a recognizer based on TT-MCTAG. To simplify the presentation, we first discuss how to extend the algorithm of fig. 4 in order to compute t-counters, and will later specify how to apply TT-MCTAG filtering conditions through such counters. The reader should however keep in mind that the two processes are strictly interleaved, with filtering conditions being tested right after the construction of each new t-counter.

We use items of the form $[\gamma_{q,r}, p_t, i, f_1,\ f_2,\ j,\ T]$, where the first six components are defined as in the case of TAG items, and the last component is a t-counter associated with the constructed derivations. Our algorithm is specified in figure 5.

The simplest case is that of rules (12) and (16). These rules do not alter the underlying derivation tree, and thus the t-counter is simply copied from the antecedent item to the consequent item.

Rules (9), (10) and (11) introduce $\gamma_{q,r}$ as the first elementary tree in the analysis ($\gamma_{q,r} \in A$ in case of rule (11)). Therefore we set the associated t-counter to $T^{(q,r)}$.

In rule (14) we substitute initial tree $\gamma_{q',r'}$ at node p in $\gamma_{q,r}$. In terms of derivation structures, we extend a derivation tree D' rooted at node v' with $Lab(v') = \gamma_{q',r'}$ to a new derivation tree D with root node v, $Lab(v) = \gamma_{q,r}$. Node v has a single child represented by the root of D'. Thus the t-counter associated with D should be $T' + T^{(q,r)}$.

A slightly different operation needs to be performed when applying rule (15). Here we have a derivation tree D with root node v, $Lab(v) = \gamma_{q,r}$ and a derivation tree D' with root node v', $Lab(v') = \gamma_{q',r'}$. When adjoining $\gamma_{q',r'}$ into $\gamma_{q,r}$, we need to add to the root of D a new child node, represented by the root of D'. This means that the t-counter associated with the consequent item should be the sum of the t-counters associated with D and D'.

Finally, rule (13) involves derivation trees D_1 and D_2, rooted at nodes v_1 and v_2, respectively. Nodes v_1 and v_2 have the same label $\gamma_{q,r}$. The application of the rule corresponds to the 'merging' of v_1 and v_2 into a new node v with label $\gamma_{q,r}$ as well, Node v inherits all of the children of v_1 and v_2. In this case the t-counter associated with the consequent item is $T_1 + T_2 - T^{(q,r)}$. Here $T^{(q,r)}$

needs to be subtracted because the contribution of tree $\gamma_{q,r}$ is accounted for in both v_1 and v_2.

We can now discuss the filtering conditions that need to be applied when using the above deduction rules. We start by observing that the algorithm in figure 5 might not even stop if there is an infinite set of derivation trees for the input string $w = a_1 \cdots a_n$ in the underlying TAG G_T. This is because each derivation can have a distinct t-counter. However, the definition of TT-MCTAG imposes that the head tree of each tuple contains at least one lexical element. Together with condition (MC), this implies that no more than n tuple instances can occur in a derivation tree for w according to G. To test for such a condition, we introduce a norm for t-counters

$$\|T\|_m \;=\; \sum_{q=1}^{|\mathcal{T}|} \max_{r=1}^{|\Gamma_q|} T[q,r]\,.$$

We then impose $\|T\|_m \leq n$ for each t-counter constructed by our deduction rule, and block the corresponding derivation if this is not satisfied.

We also need to test conditions (i) and (ii) from lemma 1. Since these conditions apply to nodes of the derivation tree, this testing is done at each deduction rule in which a consequent item may be constructed for a node $\varepsilon_\top$, that is, rules (14), (15) and (16). We introduce two specialized predicates

$$\begin{aligned} F_{\leq}(T) \;&\equiv\; \forall(q,r):\; T[q,1] \leq T[q,r]\,; \\ F_{=}(T) \;&\equiv\; \forall(q,r):\; T[q,1] = T[q,r]\,. \end{aligned}$$

We then test $F_{\leq}(T)$, which amounts to testing condition (i) for each argument tree in $A(G)$. Furthermore, if at some rule we have $F_{\leq}(T) \wedge \neg F_{=}(T)$, then we need to test for condition (ii). To do this, we consider each argument tree $\gamma_{\overline{q},\overline{r}}$, $\overline{r} \neq 1$, and compare the elementary tree $\gamma_{q,r}$ in the consequent item of the current rule with $\gamma_{\overline{q},\overline{r}}$ and $h(\gamma_{\overline{q},\overline{r}}) = \gamma_{\overline{q},1}$, to select the appropriate subcondition of (ii).

As an example, assume that we are applying rule (15) as in figure 5, with $p = \varepsilon$. Let $T_c = T + T'$ be the t-counter associated with the consequent item. When we come to process some argument tree $\gamma_{\overline{q},\overline{r}}$ such that $T_c[\overline{q},\overline{r}] - T_c[\overline{q},1] > 0$ and $\gamma_{q,r} \notin \{\gamma_{\overline{q},\overline{r}}, \gamma_{\overline{q},1}\}$, we need to test (ii)c. This is done by requiring

$$T'[\overline{q},\overline{r}] - T'[\overline{q},1] = T_c[\overline{q},\overline{r}] - T_c[\overline{q},1].$$

If we are instead applying rule (16) with $p = \varepsilon$ and $T[\overline{q},\overline{r}] - T[\overline{q},1] > 0$, then we test (ii)a, since there is no adjunction at the root node, by requiring $\gamma_{q,r} = \gamma_{\overline{q},\overline{r}}$ and $T[\overline{q},\overline{r}] - T[\overline{q},1] = 1$.

We block the current derivation whenever the conditions in lemma 1 are not satisfied.

The algorithm recognizes w if and only if some item $[\gamma_{q,1}, \varepsilon_\top, 0, -, -, n, T]$ can be inferred satisfying $\gamma_{q,1} \in I$, $Lab(\gamma_{q,1}, \varepsilon) = S$ and $F_{=}(T)$. The correctness immediately follows from the correctness of the underlying TAG parser and from lemma 1.

Finally, we turn to the computational analysis of the algorithm. We assume a tabular implementation of the process of item inference using our deduction rules. Our algorithm clearly stops after some finite amount of time, because of the filtering condition $\|T\|_m \leq n$. We then need to derive an upper bound on the number of applications of deduction rules. To do this, we use an argument that is rather standard in the tabular parsing literature. The number of t-counters satisfying $\|T\|_m \leq n$ is $\mathcal{O}(n^{c_G})$, with $c_G = \sum_{i=1}^{|\mathcal{T}|} |\Gamma_i|$. Since all of the other components in an item are bounded by $\mathcal{O}(n^4)$, there are polynomially (in n) many items that can be constructed for an input w. It is not difficult to see that each individual item can be constructed by a number of rule applications bounded by a polynomial as well. Therefore, the total number of applications of our deduction rules is also bounded by some polynomial in n. We thus conclude that the languages generated by the class TT-MCTAG are all included in PTIME.

5 Conclusion and open problems

We have shown in this paper that the class of languages generated by TT-MCTAG is included in PTIME, by characterizing the definition of TT-MCTAG through some conditions that can be tested locally. PTIME is one of the required properties in the definition of the class of Mildly Context-Sensitive (MCS) formalisms (Joshi et al., 1991). In order to settle membership in MCS for TT-MCTAG, what is still missing is the constant-growth property or, more generally, the semilinearity property.

Acknowledgments

The work of the first author has been supported by the DFG within the Emmy-Noether Program. The second author has been partially supported by MIUR under project PRIN No. 2007TJNZRE_002.

References

Tilman Becker, Aravind K. Joshi, and Owen Rambow. 1991. Long-distance scrambling and tree adjoining grammars. In *Proceedings of ACL-Europe*.

Tilman Becker, Owen Rambow, and Michael Niv. 1992. The Derivationel Generative Power of Formal Systems or Scrambling is Beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

Lucas Champollion. 2007. Lexicalized non-local MC-TAG with dominance links is NP-complete. In Gerald Penn and Ed Stabler, editors, *Proceedings of Mathematics of Language (MOL) 10*, CSLI On-Line Publications.

Joan Chen-Main and Aravind Joshi. 2007. Some observations on a graphical model-theoretical approach and generative models. In *Model Theoretic Syntax at 10. Workshop, ESSLLI 2007*, Dublin, Ireland.

Armin B. Cremers and Otto Mayer. 1973. On matrix languages. *Information and Control*, 23:86–96.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoning Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–123. Springer, Berlin.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Science*, 10:136–163.

A. Joshi, K. Vijay-Shanker, and D. Weir. 1991. The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. Shieber, and T. Wasow, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge MA.

Aravind K. Joshi. 1985. Tree adjoining grammars: How much contextsensitivity is required ro provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press.

Laura Kallmeyer and Yannick Parmentier. 2008. On the relation between Multicomponent Tree Adjoining Grammars with Tree Tuples (TT-MCTAG) and Range Concatenation Grammars (RCG). In Carlos Martín-Vide, Friedrich Otto, and Henning Fernaus, editors, *Language and Automata Theory and Applications. Second International Conference, LATA 2008*, number 5196 in Lecture Notes in Computer Science, pages 263–274. Springer-Verlag, Heidelberg Berlin.

Laura Kallmeyer. 2005. Tree-local multicomponent tree adjoining grammars with shared nodes. *Computational Linguistics*, 31(2):187–225.

Timm Lichte and Laura Kallmeyer. 2008. Factorizing Complementation in a TT-MCTAG for German. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, pages 57–64, Tübingen, June.

Timm Lichte. 2007. An MCTAG with Tuples for Coherent Constructions in German. In *Proceedings of the 12th Conference on Formal Grammar 2007*, Dublin, Ireland.

Rebecca Nesson and Stuart Shieber. 2008. Synchronous Vector TAG for Syntax and Semantics: Control Verbs, Relative Clauses, and Inverse Linking. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, Tübingen, June.

Owen Rambow and Giorgio Satta. 1992. Formal properties of non-locality. In *Proceedings of 1st International Workshop on Tree Adjoining Grammars*.

Owen Rambow and Giorgio Satta. 1994. A rewriting system for free word order syntax that is non-local *and* mildly context sensitive. In C. Martín-Vide, editor, *Current Issues in Mathematical Linguistics*, North-Holland Linguistic series, Volume 56. Elsevier-North Holland, Amsterdam.

Owen Rambow, K. Vijay-shanker, and David Weir. 1995. Parsing d-Ttree grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies, Prague*, pages 252–259.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania.

Giorgio Satta. 1995. The membership problem for unordered vector languages. In *Developments in Language Theory*, pages 267–275.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and Implementation of Deductive Parsing. *Journal of Logic Programming*, 24(1&2):3–36.

Anders Søgaard, Timm Lichte, and Wolfgang Maier. 2007. The complexity of linguistically motivated extensions of tree-adjoining grammar. In *Recent Advances in Natural Language Processing 2007*, Borovets, Bulgaria.

K. Vijay-Shanker and Aravind K. Joshi. 1985. Some computational properties of Tree Adjoining Grammars. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 82–93.

K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In 25th *Meeting of the Association for Computational Linguistics (ACL'87)*.

David J. Weir. 1988. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania.